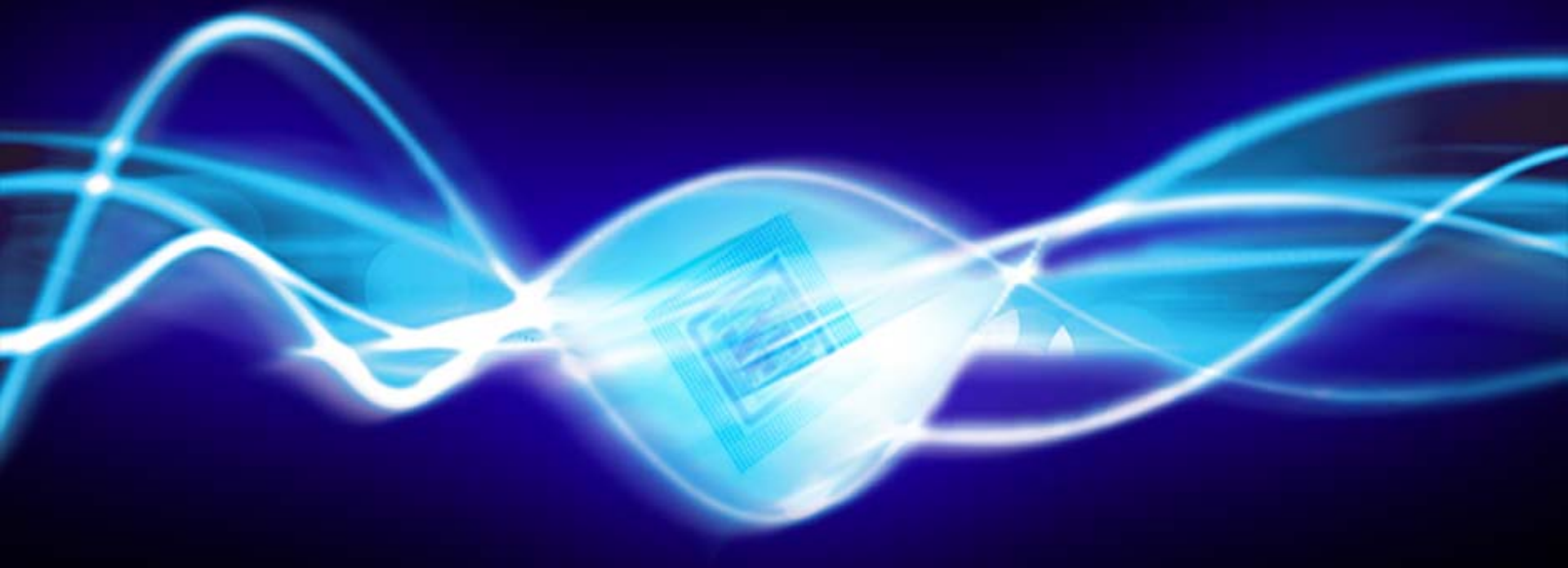
MIPS TECHNOLOGIES

At the core of the user experience.®

The MIPS32® 34K™ Processor: Ultimate Design Flexibility for Embedded Applications



Ryan Kinter, Lead Microarchitect

Hotchips 18, August 22, 2006

Agenda

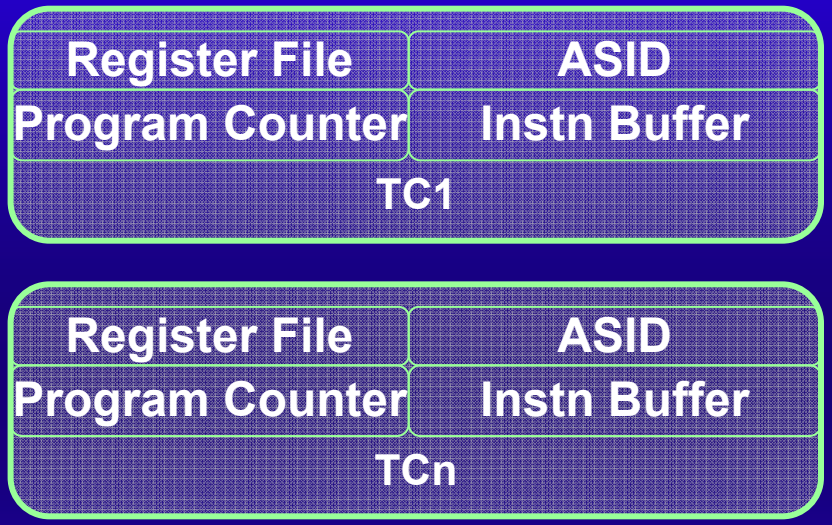
- 34K core multi-threaded architecture
- Applications for multi-threading
- Microarchitectural features
- 34K core specs

MIPS32® 34K™ Processor Core Family

- Based on the proven 24K®/24KE™ core pipeline
- First implementation of MIPS' multithreading architecture
- Includes DSP extensions
- First released in October 2005
- Working silicon running 5 threads in May 2006

- Supports configuration options of the 24KE core plus MT specific options
  - Customizable policy manager & gating storage modules
  - 1 or 2 virtual processing elements
  - 1 to 9 thread contexts

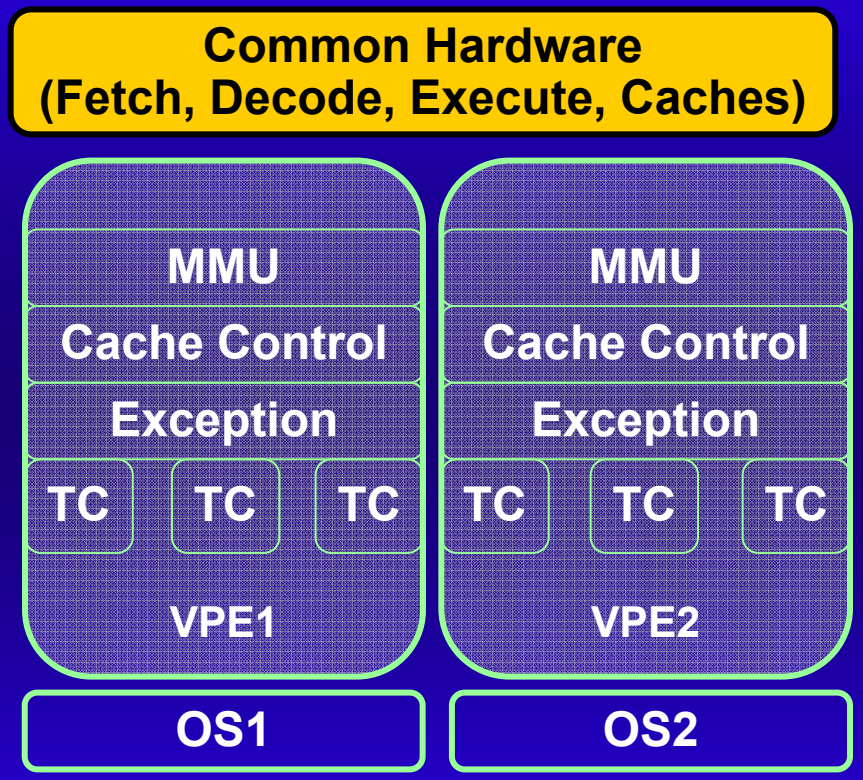
Thread Contexts



- The hardware associated with a software thread
- A TC executes instructions
- Includes user state
- 34K core can have 1 to 9 TCs
  - Minimal area per TC

- Can dedicate TCs to specific, critical tasks
  - Great for embedded applications

Virtual Processing Elements



- Contains system coprocessor state
- With a TC, looks like a complete MIPS32 processor to software
- 34K core can have 1 or 2 VPEs

- Each VPE can run an independent operating system
  - 2 OSes without virtualization software

Multi-threading Benefits

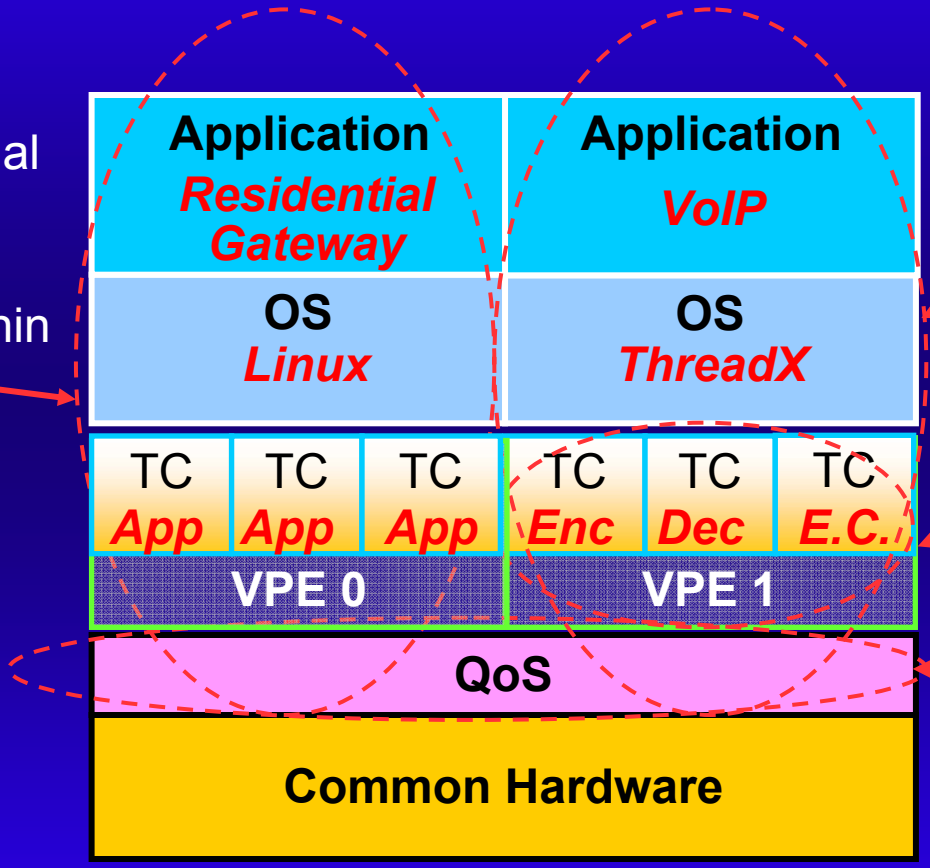
- Fills short-term pipeline bubbles
- Runs other threads during memory accesses
  - Improves performance by increasing IPC
- Removes overhead from switching threads
  - Improves performance by reducing instruction count
  - Reduces response time

Application: 1-1 mapping of processes

- In an RTOS with few processes, each one can have its own thread context.
  - Never need to do a context switch!
  - Reduces the number of instructions that need to be executed
- For example, processing parallel data streams
  - 1 control TC + up to 8 data TCs
- If there is an unknown number of processes or more than the number of TCs, hybridize:
  - Dedicate hardware for critical tasks
  - Switch processes in and out on remaining TCs.

Example System

Multi-tasking OS running residential gateway applications on multiple TCs within one VPE



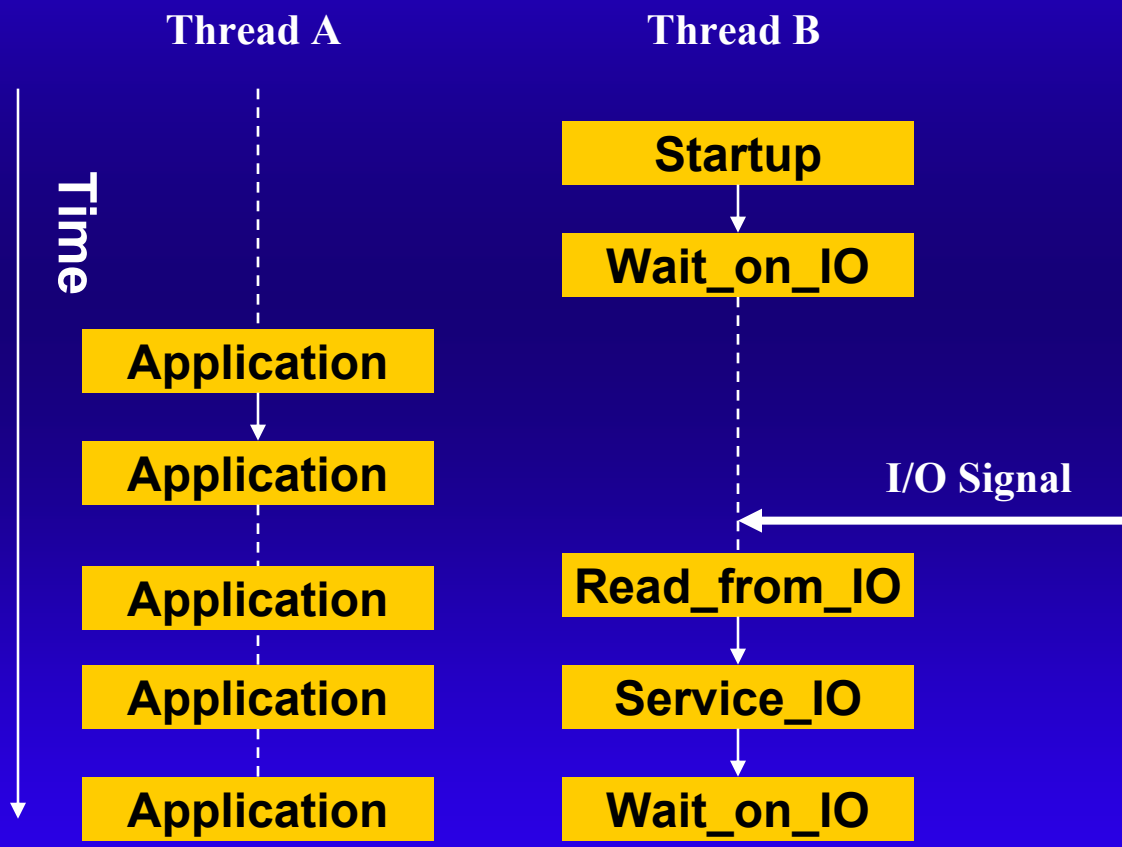
RTOS handling VoIP on other VPE

Dedicated TCs for each function

Policy manager sets relative priority of the tasks

Application: Interrupt Service Thread

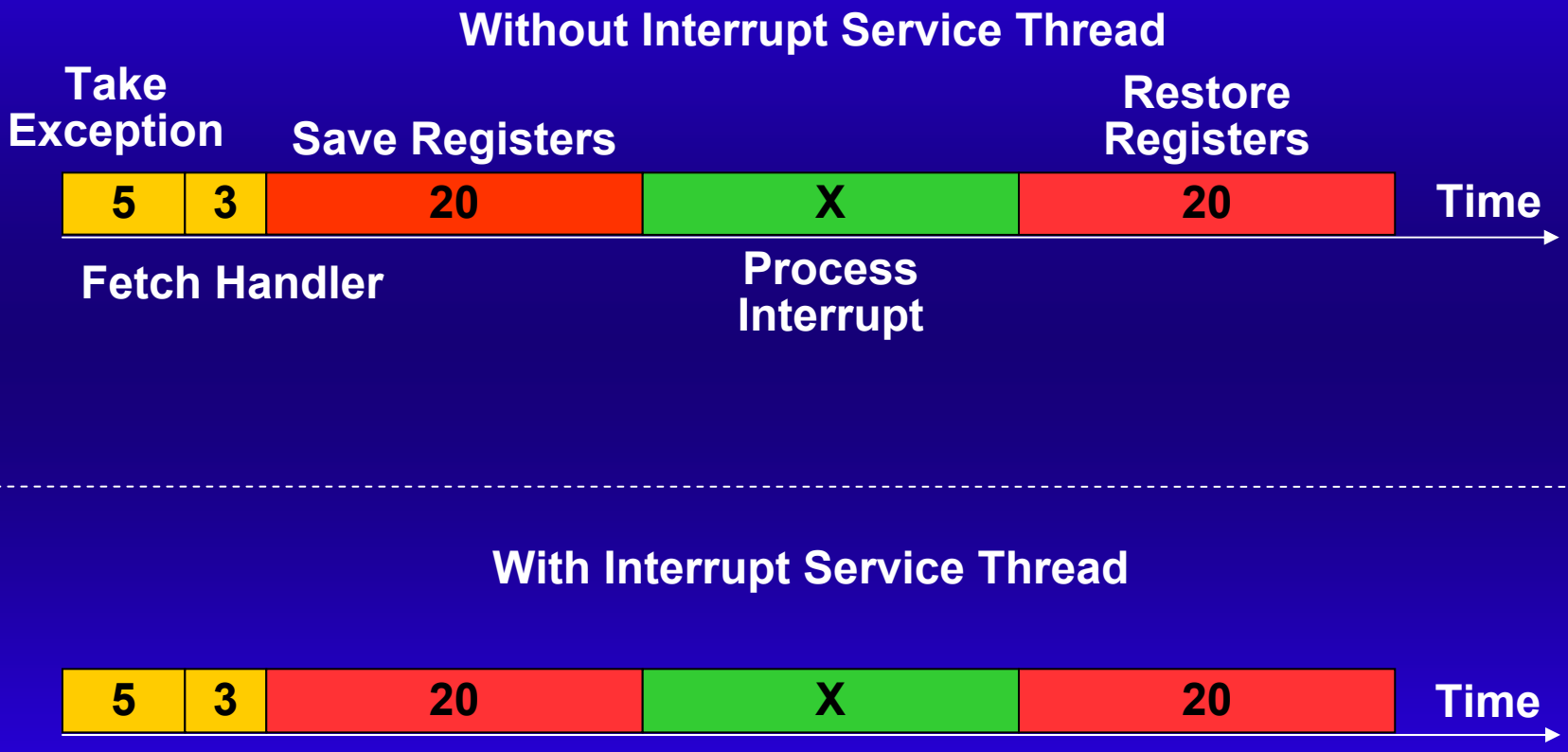
- Allocate TCs to service system interrupts
- Ready to run immediately
- "Zero" interrupt latency



Interrupt service threads

- YIELD instruction specifies that the thread should wait until a particular system event occurs
- Following instructions are prefetched and stored in an instruction buffer
- When system event occurs, the thread can resume execution.
- TC has a dedicated register file, so there is no need for a context save or restore

Interrupt service threads



Interrupt Service Threads

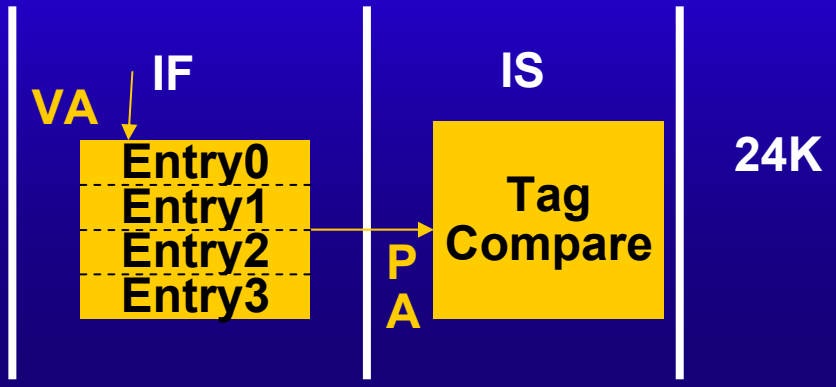
- Thread scheduling policies can set priority of service thread versus other threads
- Could have multiple interrupt service threads for various interrupt sources
- Can be generalized to other types of tasks
  - Periodic timer triggering profiling task
  - I/O coherence request

How to support 9 TCs?

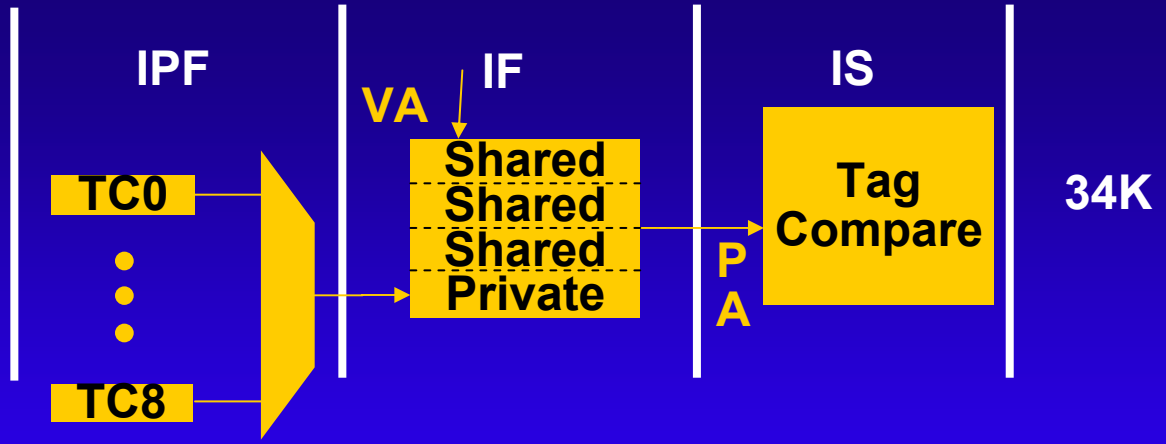
- Wide range of configurations requires novel solutions to have an attractive product with larger configurations as well as the more typical small ones
  - Cannot have a large area or speed penalty for adding TCs
  - Cannot sacrifice performance
- Need to be selective about when to replicate resources per TC

Example: Micro TLB

- 24K core had 4 entry ITLB
- Need more entries for 9 TCs
- Larger array would impact speed

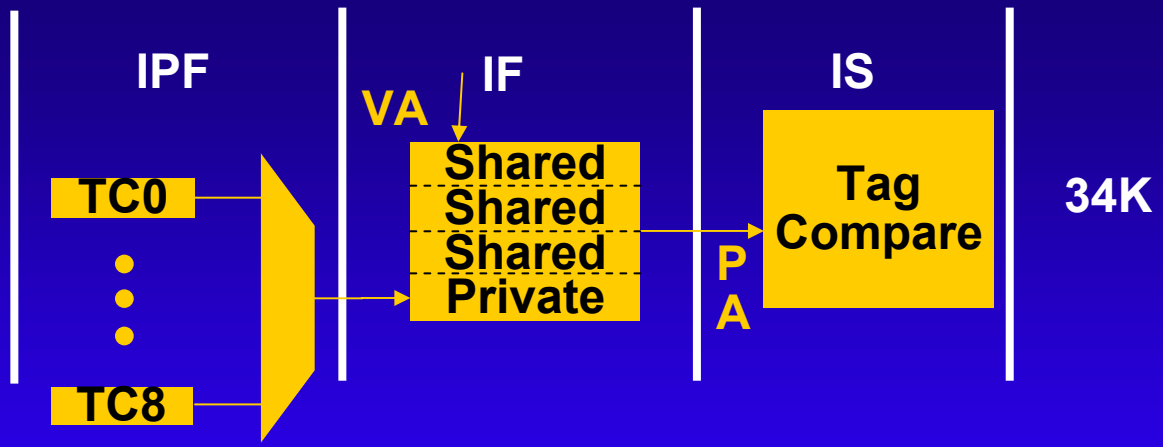


- 1 private entry per TC, plus 3 shared



Example: Micro TLB

- Lookup is still across 4 entries, no speed penalty
- When a single TC is running, it will have access to 4 entries
  - Same as it would have on a 24K core
- Dedicated entry simplifies control logic

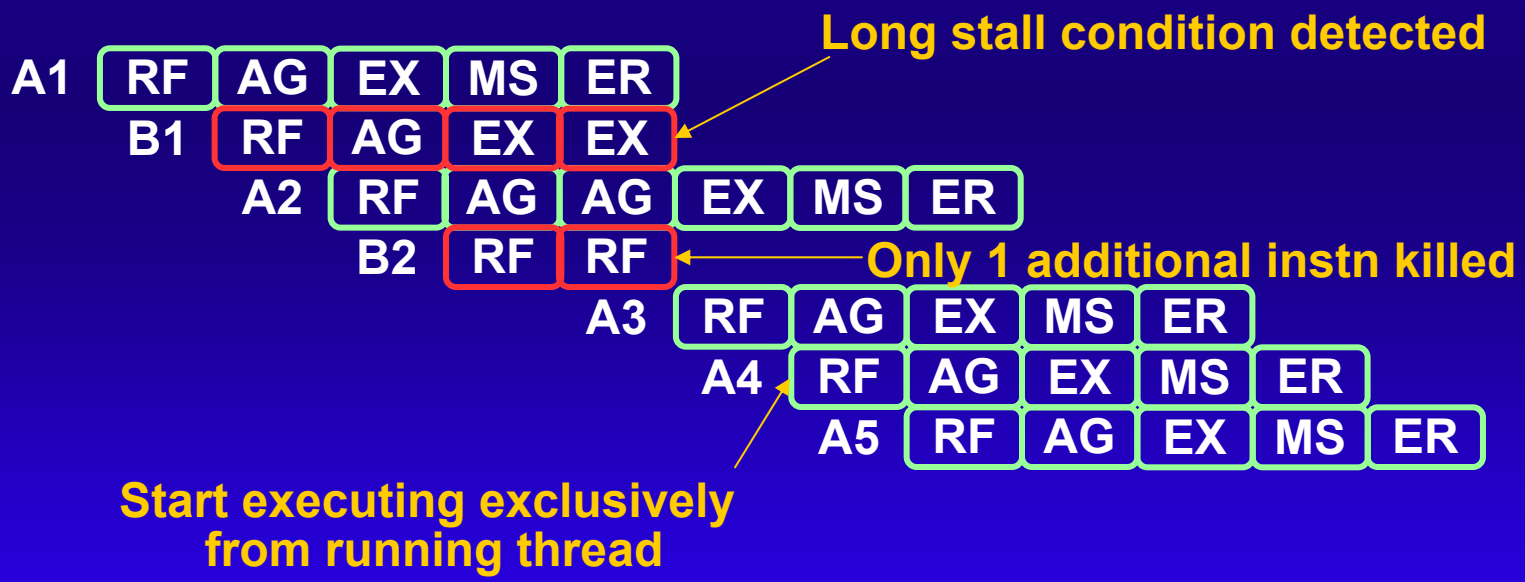


Example: Instruction Buffers

- 24K core features 8 entry instruction buffer
  - Decouples the fetch unit from the execution unit
  - Fetch unit executes ahead to try to keep the buffer full
- Replicated per TC on 34K core
- Instruction buffer still decouples the fetch unit from the execution unit
- Also isolates the fetch unit from the policy manager

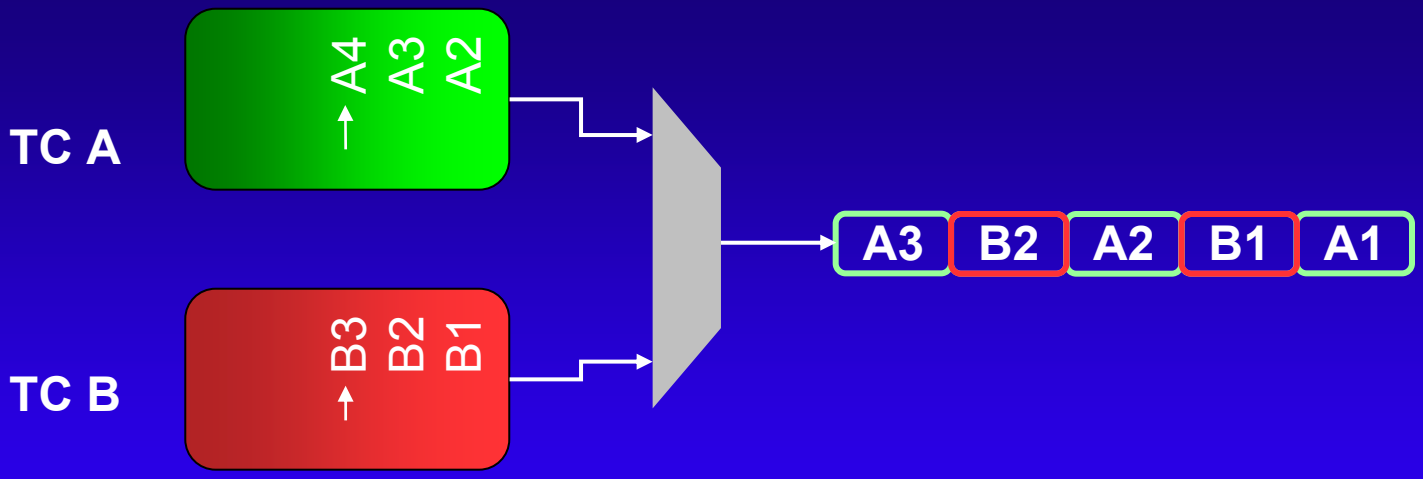
Instruction Buffers

- Replicating moves the thread selection point further down the pipe
  - Core can respond faster
  - Reduces the number of instructions killed



Instruction Buffers

- Instruction buffer also acts as a skid buffer
  - Dispatched instructions are held in buffer
  - Stalling instructions can be removed from the pipeline
    - Move buffer pointer to restore



Load Data Queue

- Buffer in Load/Store Unit
- Holds load miss info
- 4 entries on 24K core
- Option to increase to 9 entries on 34K core
- Enough for 1 per TC for gating storage
- Shared resource

34K™ Core Specs

- Speed: 500MHz
  - TSMC 90G, nominal V_t
  - Worst-case PVT
- Area: 5.1mm^2 with 2 VPEs, 4 TCs (Including 32KB/32KB caches)
  - 0.1 - 0.2mm^2 for a TC
  - 0.2 - 0.3mm^2 for a VPE
- Power: 0.59 mW/MHz @1.0V
  - Running four instances of Dhrystone
- Licensable now
  - First released in October 2005
  - Working silicon running 5 threads

Summary

- 34K core features multi-threading
  - Wide range of applications
- Dedicated hardware threads
  - Reduce instruction count
  - Improve response time
- Low thread cost